Text Editing on Unix

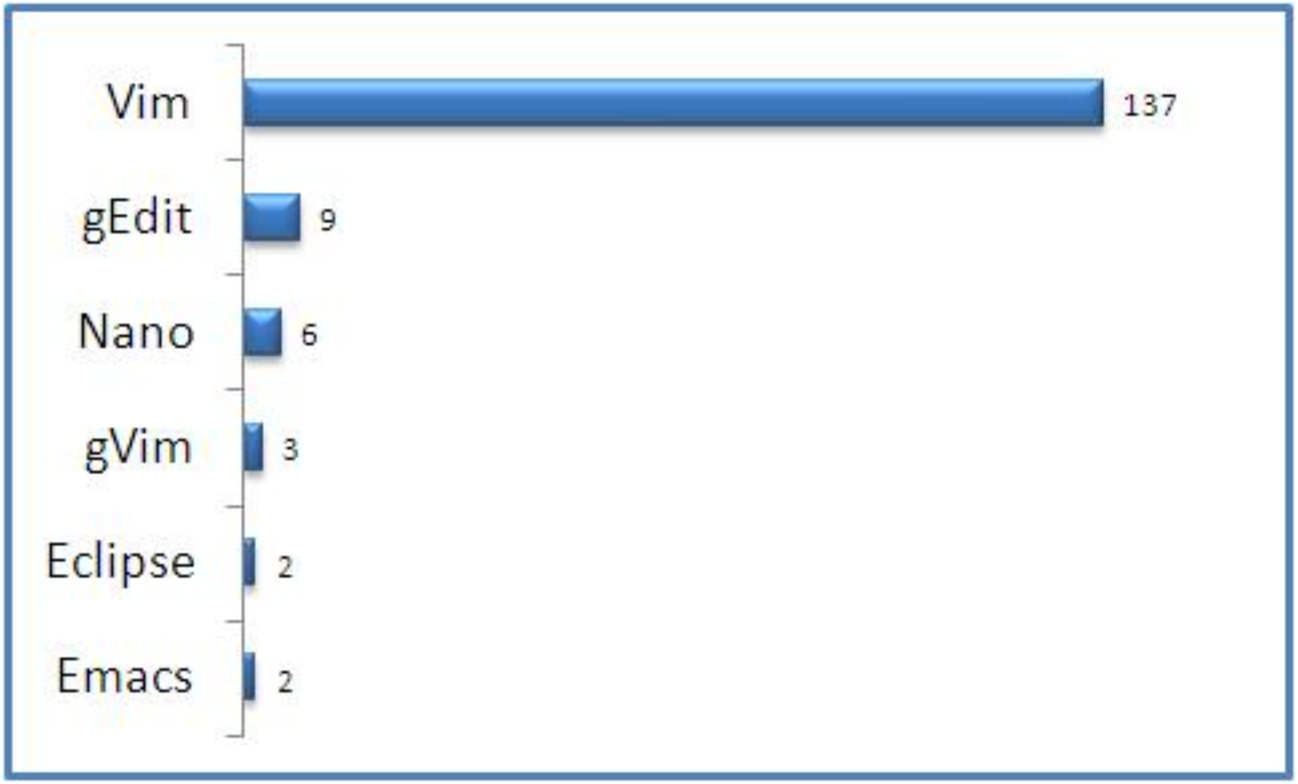
Shiquan Su
National Institute for Computational Science
User Assistant Group

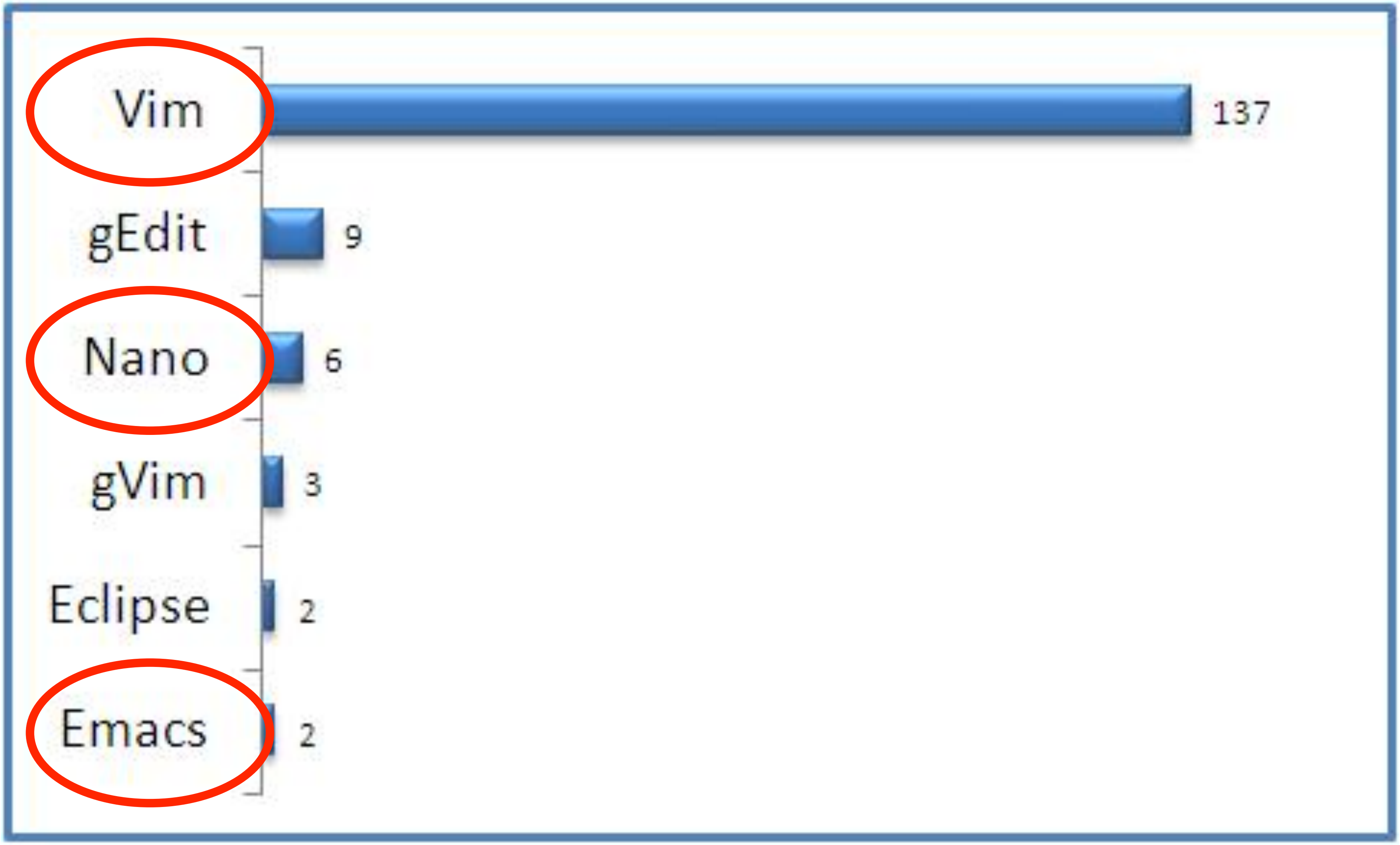
THE GEEK STUFF

Linux | DB | Open Source | Web

As part of the contest that was conducted a while back, I got around 200 responses from the geeky readers who choose their favorite Linux text editor.

Based on this data, the top spot in the best Linux text editor goes to…

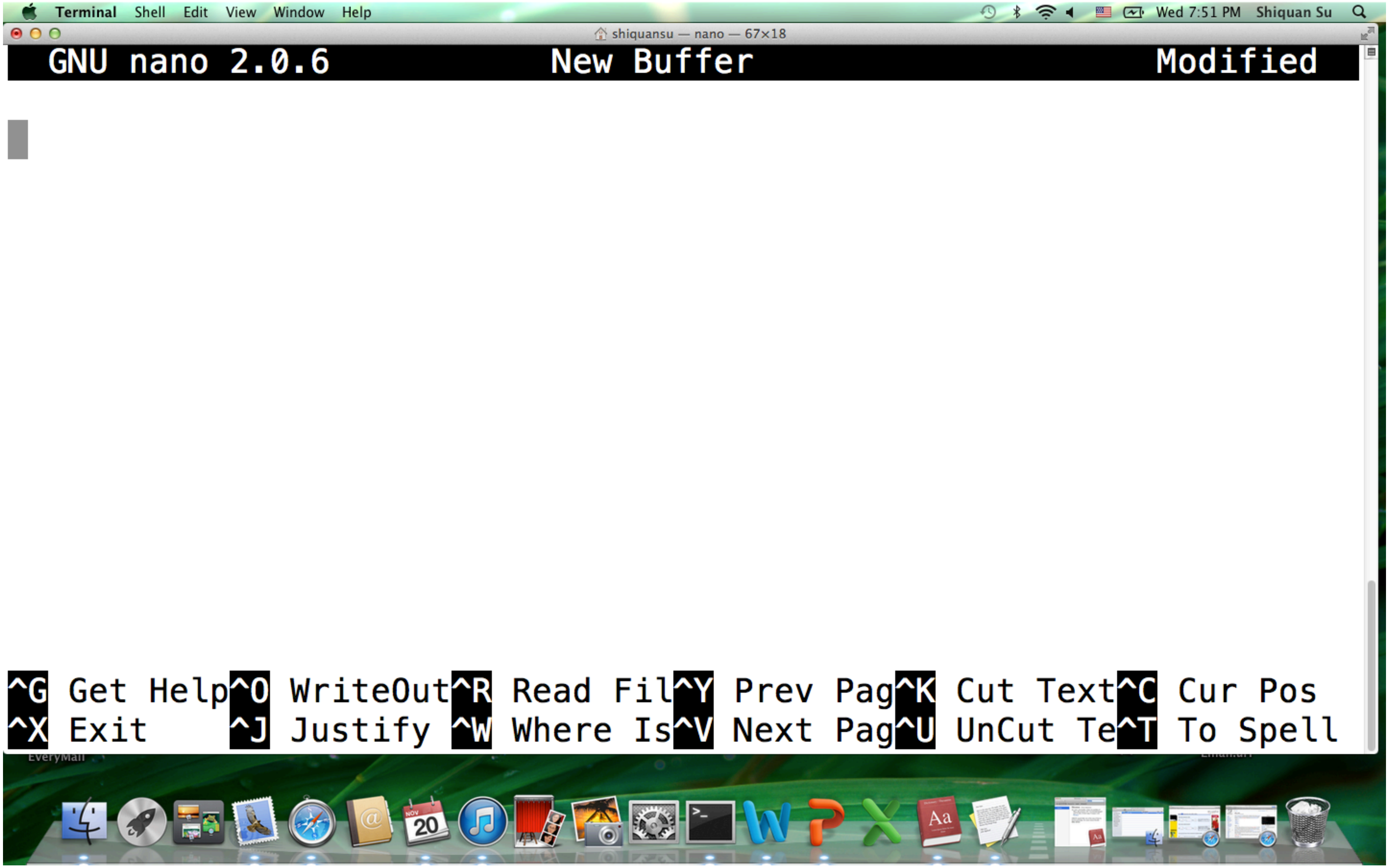



Vim
137
gEdit
9
Nano
6
gVim
3
Eclipse
2
Emacs
2

Set up a text example to try different text editors:

1, google: mpi c++ hello world example
2, choose the second one from “UMBC”
3, copy and paste the source code to example.cpp

GNU nano



```
  GNU nano 2.0.6              New Buffer                     Modified

^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

History:

nano was first created in 1999 with the name *TIP* (*This isn't Pico*), by Chris Allegretta. His motivation was to create a free software replacement for Pico, which was not distributed under a free software license. The name was changed to nano on January 10, 2000 to avoid a naming conflict with the existing Unix utility tip. The name comes from the system of SI prefixes, in which nano is 1000 times larger than pico, In February 2001, nano became a part of the GNU Project.

nano implements some features that Pico lacks, including colored text, regular expression search and replace, smooth scrolling, multiple buffers, rebindable key support, and (experimental) undoing and redoing of edit changes.

On August 11, 2003, Chris Allegretta officially handed the source code maintenance for nano to David Lawrence Ramsey. On December 20, 2007, Ramsey stepped down as nano's maintainer.

Control keys:

nano, like Pico, is keyboard-oriented, controlled with control keys. For example, ⎈ Ctrl+O saves the current file; ⎈ Ctrl+W goes to the search menu. Nano puts a two-line "shortcut bar" at the bottom of the screen, listing many of the commands available in the current context. For a complete list, ⎈ Ctrl+G gets the help screen.

Unlike Pico, nano uses meta keys to toggle its behavior. For example, ◆ Meta+S toggles smooth scrolling mode on and off. Almost all features that can be selected from the command line can be dynamically toggled.

For example, the Meta key on MacBook Pro is the ‘esc’ key.

Command list (1/4):

^G	(F1)		Display this help text
^X	(F2)		Close the current file buffer / Exit from nano
^O	(F3)		Write the current file to disk
^R	(F5)		Insert another file into the current one
^W	(F6)		Search for a string or a regular expression
^Y	(F7)		Move to the previous screen
^V	(F8)		Move to the next screen
^K	(F9)		Cut the current line and store it in the cutbuffer
^U	(F10)		Uncut from the cutbuffer into the current line
^C	(F11)		Display the position of the cursor
^T	(F12)		Invoke the spell checker, if available
^_	(F13)	(M-G)	Go to line and column number
^\	(F14)	(M-R)	Replace a string or a regular expression
^^	(F15)	(M-A)	Mark text at the cursor position
	(F16)	(M-W)	Repeat last search

Command list (2/4):

M-^	(M-6)	Copy the current line and store it in the cutbuffer

^F		Move forward one character
^B		Move back one character
^Space		Move forward one word
M-Space		Move back one word
^P		Move to the previous line
^N		Move to the next line

^A		Move to the beginning of the current line
^E		Move to the end of the current line
M-(	(M-9)	Move to the beginning of the current paragraph
M-)	(M-0)	Move to the end of the current paragraph
M-\	(M-|)	Move to the first line of the file
M-/	(M-?)	Move to the last line of the file

Command list (3/4):

M-]		Move to the matching bracket
M--	(M-_)	Scroll up one line without scrolling the cursor
M-+	(M-=)	Scroll down one line without scrolling the cursor

M-<	(M-,)	Switch to the previous file buffer
M->	(M-.)	Switch to the next file buffer

M-V		Insert the next keystroke verbatim
^I		Insert a tab at the cursor position
^M		Insert a newline at the cursor position
^D		Delete the character under the cursor
^H		Delete the character to the left of the cursor
M-T		Cut from the cursor position to the end of the file

M-D		Count the number of words, lines, and characters
^L		Refresh (redraw) the current screen

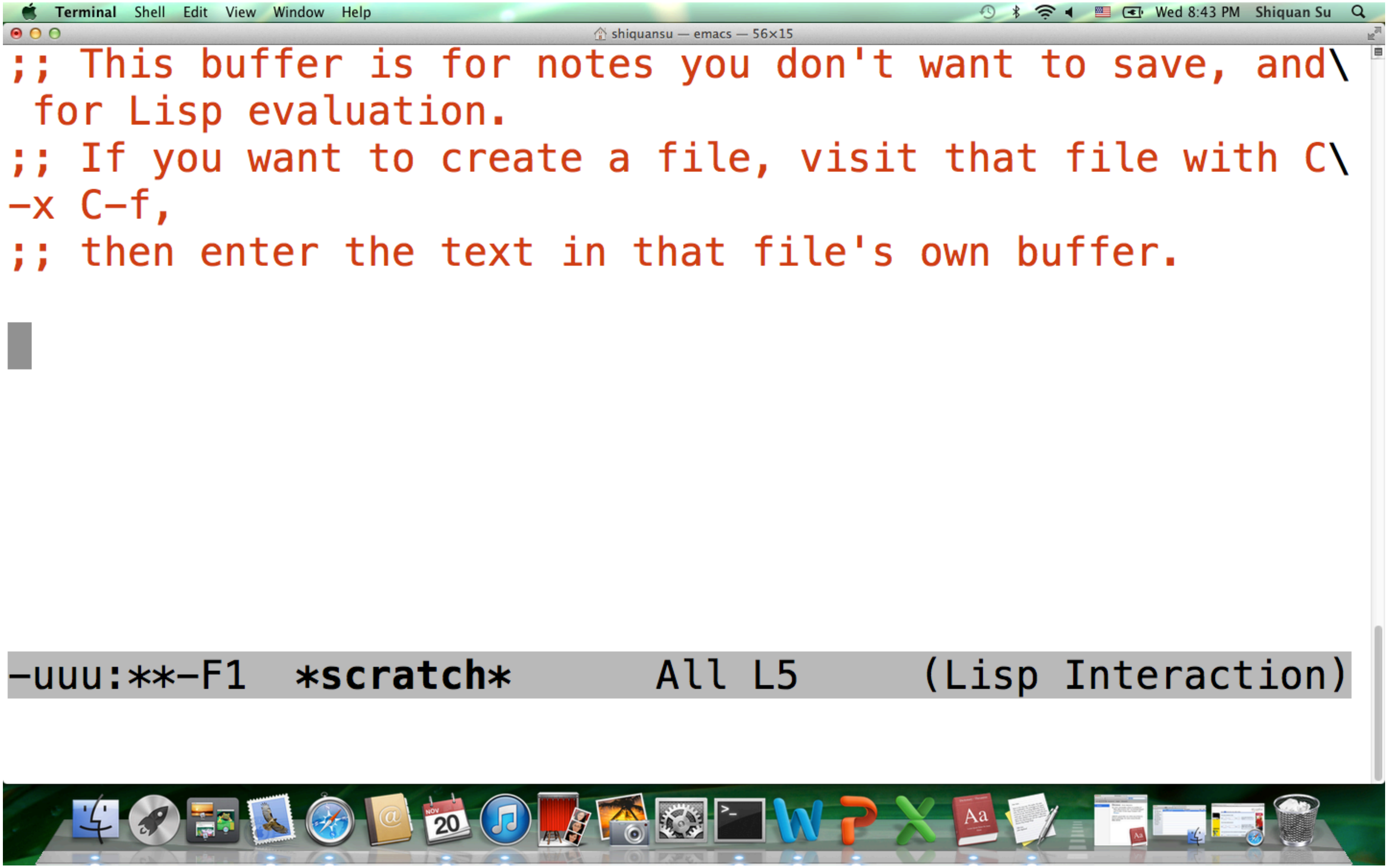
Command list (4/4):

M-X	Help mode enable/disable
M-C	Constant cursor position display enable/disable
M-O	Use of one more line for editing enable/disable
M-S	Smooth scrolling enable/disable
M-P	Whitespace display enable/disable
M-Y	Color syntax highlighting enable/disable

M-H	Smart home key enable/disable
M-I	Auto indent enable/disable
M-K	Cut to end enable/disable
M-L	Long line wrapping enable/disable
M-Q	Conversion of typed tabs to spaces enable/disable

M-B	Backup files enable/disable
M-F	Multiple file buffers enable/disable
M-M	Mouse support enable/disable
M-N	No conversion from DOS/Mac format enable/disable
M-Z	Suspension enabled/disabled

GNU Emacs



```
;; This buffer is for notes you don't want to save, and\
 for Lisp evaluation.
;; If you want to create a file, visit that file with C\
-x C-f,
;; then enter the text in that file's own buffer.



-uuu:**-F1  *scratch*      All L5     (Lisp Interaction)
```

GNU Emacs is an extensible, customizable text editor—and more. At its core is an interpreter for Emacs Lisp, a dialect of the Lisp programming language with extensions to support text editing. The features of GNU Emacs include:

- Content-sensitive editing modes, including syntax coloring, for a variety of file types including plain text, source code, and HTML.
- Complete built-in documentation, including a tutorial for new users.
- Full Unicode support for nearly all human languages and their scripts.
- Highly customizable, using Emacs Lisp code or a graphical interface.
- A large number of extensions that add other functionality, including a project planner, mail and news reader, debugger interface, calendar, and more. Many of these extensions are distributed with GNU Emacs; others are available separately.

Help Commands

* **C-h** ***help-command****: first character in lots of useful help commands*

* **C-h t** ***help-with-tutorial****: command to run the tutorial*

C-h i ***information****: describes most of the emacs commands in man style pages*

C-h k ***describe-key****: tells you what a particular key stroke does*

* **C-h a** ***command-apropos****: prompts for a string and then searches for all emacs commands that contains that string*

ESC ? *also does* ***command-apropos***

* **C-h ?** ***help-for-help****: describes how to use the help facilities*

File Reading and Writing Commands

*   **C-x C-f** ***find-file****: first prompts for a filename and then loads that file into a editor buffer of the same name*

*   **C-x C-s** ***save-buffer****: saves the buffer into the associated filename*

C-x C-w ***write-named-file****: prompts for a new filename and writes the buffer into it*

Cursor/Screen Movement Commands

*	**C-a**	*move cursor to (at) beginning-of-line*
	C-e	*move cursor to end-of-line*
*	**C-f**	*move cursor forward one character*
*	**C-b**	*move cursor backward one character*
*	**C-n**	*move cursor to next line*
*	**C-p**	*move cursor to previous line*
	C-v	*scroll file forward by one screenful*
	ESC v	*scroll file backward by one screenful*
*	**ESC <**	*go to beginning-of-buffer*
*	**ESC >**	*go to end-of-buffer*
	ESC f	*move cursor forward one word*
	ESC b	*move cursor backward one word*

Copy and Delete Commands

C-d ***delete-char***: *delete character under cursor*

ESC d ***delete-word***: *delete from cursor to end of word immediately ahead of the cursor*

* **C-k** ***kill-line***: *delete the rest of the current line*

* **C-@** ***set-mark-command***: *mark is used to indicate the beginning of an area of text to be yanked*

* **C-w** ***kill-region***: *delete the area of text between the mark and the current cursor position*

* **C-y** ***yank***: *insert at current cursor location whatever was most recently deleted*

ESC w ***copy-region-as-kill***: *copy area between mark and cursor into kill-buffer so that it can be yanked into someplace else*

Search Commands

* **C-s** ***isearch-forward***: *prompts for text string and then searches from the current cursor position forwards in the buffer*

C-r ***isearch-backward***: *like* ***isearch-forward***, *but searches from the current cursor position to end of buffer for text string*

ESC % ***query-replace***: *prompts for a search string and a string with which to replace the search string*

Window and Buffer Commands

C-x 0 ***zero-window***: *deletes current window*

C-x 2 ***double-window***: *splits current window into two parts,*
allowing you to edit at two different locations in the same file
or permitting you to view two different files at the same time

C-x b ***switch-to-buffer***: *display a different buffer on the screen*

C-x o ***other-window***: *move the cursor to the other window*
(assuming that you have two windows/buffers open at once

* **C-x C-b** ***list-buffers***: *lists those buffers currently loaded into emacs*

Exiting Emacs, Fixing Mistakes and Other Important Stuff

- **C-x C-c** ***save-buffers-kill-emacs****: when you are finished editing, to save the edited but unsaved buffers and to return you to the UNIX prompt*

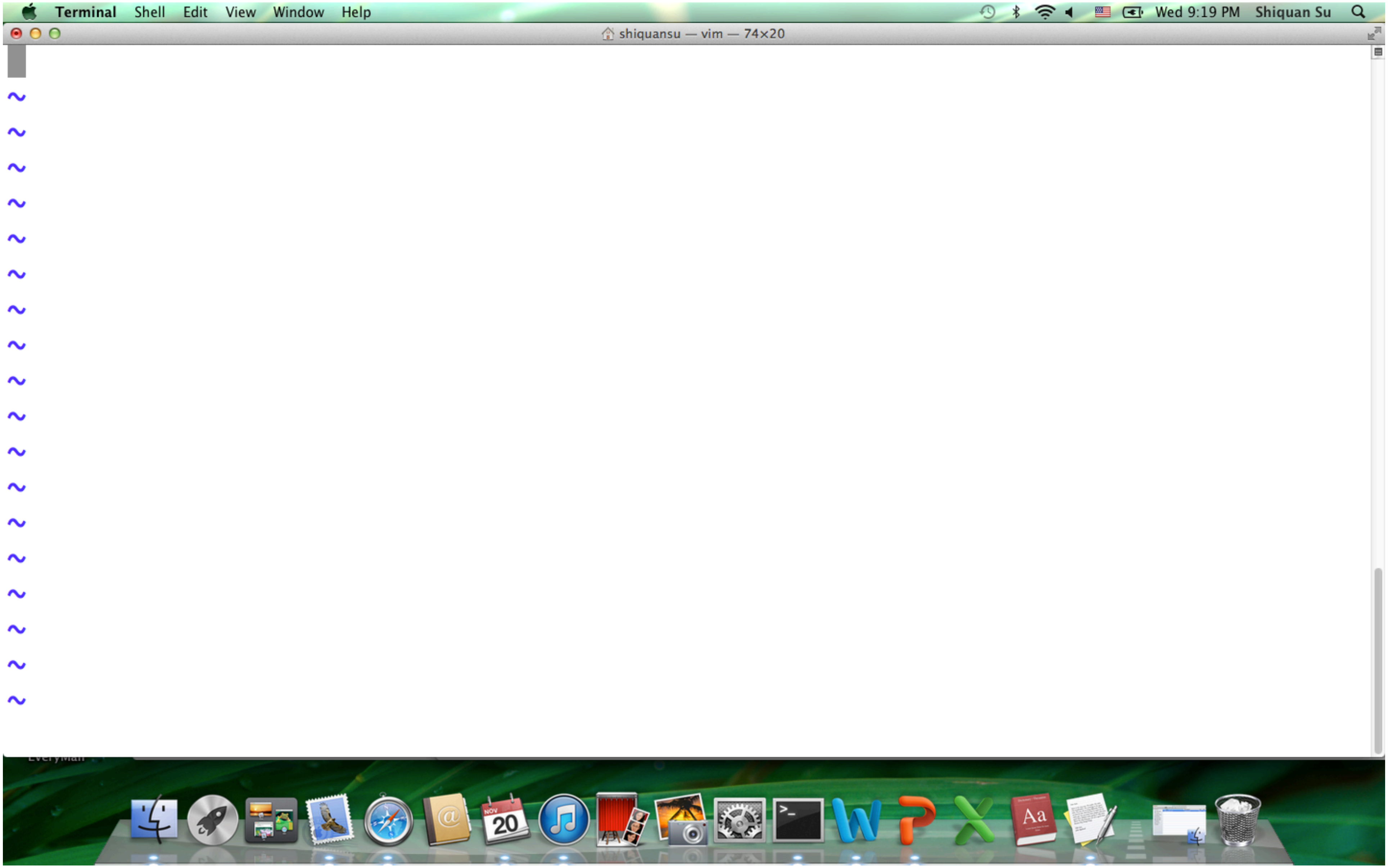
  C-g ***keyboard-quit****: if while typing a command you make a mistake and want to stop, this aborts a command in progress*

  C-u ***universal-argument****: if you want to do a command several times, type this command followed by a number (for the number of times) followed by the command you wish repeated*

- **C-x u** ***undo****: undoes the last command typed, in case you made a mistake*

  ESC x ***execute-extended-command****: prompts for the name of an emacs command; allows you to execute a command if you know roughly what it is called but cannot remember the key strokes for it*

vi


Terminal
Shell
Edit
View
Window
Help
Wed 9:19 PM
Shiquan Su
shiquansu — vim — 74×20

```

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

About VI

vi is a screen-oriented text editor originally created for the Unix operating system. The portable subset of the behavior of vi and programs based on it, and the ex editor language supported within these programs, is described by (and thus standardized by) the Single Unix Specification and POSIX.

Over the years since its creation, vi became the de facto standard Unix editor and a nearly undisputed number one editor until the rise of Emacs after about 1984. The Single UNIX Specification specifies vi, so every conforming system must have it.

A 2009 survey of Linux Journal readers found that vi was the most widely used text editor among respondents, beating gedit, the second most widely used editor by nearly a factor of two (36% to 19%).

VI editor

- Used to create a file
- 2 modes: Insert and View
- Press ESC to be in View mode
- Press letter “i” to be in insert mode
- To save your work press ESC and “:wq”
- To quit without saving press ESC and “:q!”
- Webpage for help : google “vi editor summary pdf” or www.amath.colorado.edu/computing/unix/vi/

Starting vi – the vi material are copied from the webpage

Opening an existing file

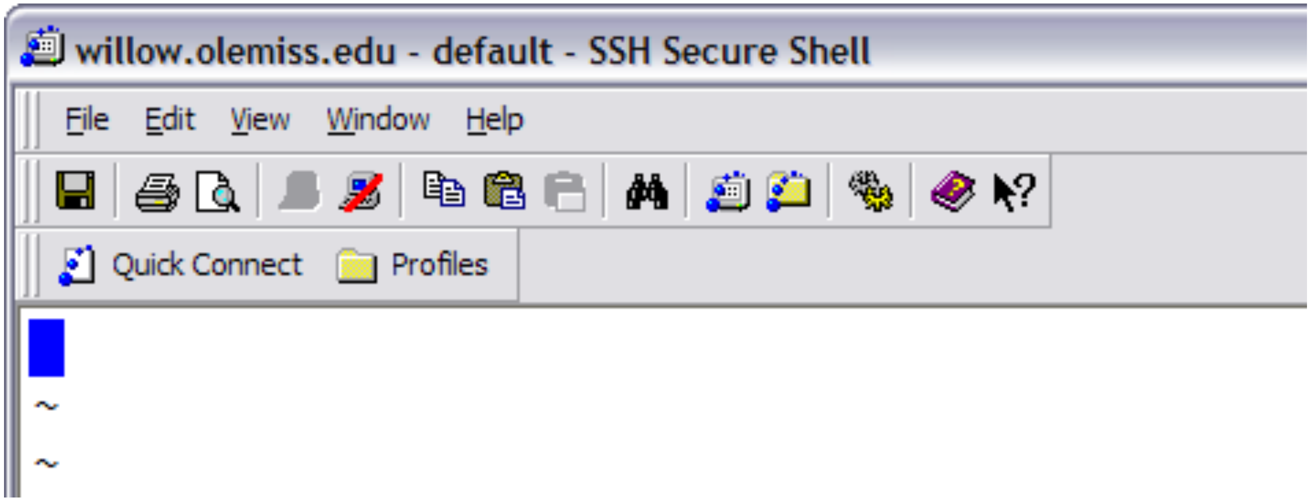
vi *filename*

Creating a new file

vi *filename*

In your workshop directory, create a new file called ***mysong***

```
vi mysong
```



Vi Modes of Operation

- **Command Mode**

  Allows the entry of commands to manipulate text

  Default mode when vi starts

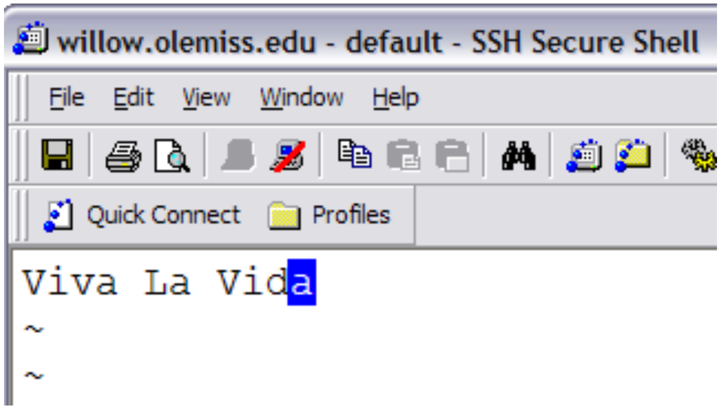
  Use Escape key to move into command mode

- **Insert Mode and**

  Puts anything you type into the current file

  To get into insert mode, commands are
  ***a* (append) and *i* (insert)**

*1. Use the **i command** to move into insert mode (**Press i key**).*
2. Attempt to type in the title of your favorite song.
*3. Use the **Esc key** to move to command mode.*



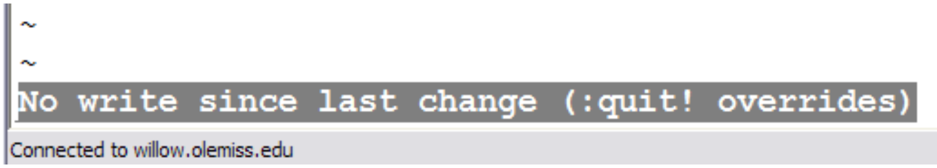
```
Viva La Vida
~
~
```

Exiting the Vi Editor

:q **Quit the editor**

:q! **Quit without saving changes to the file**

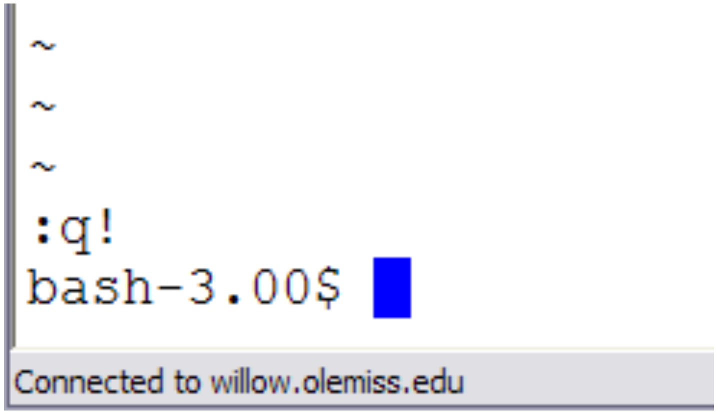
1. *Use the **Esc key** to make sure you are in command mode.*
2. *Use the **:q** command to try to quit vi*



```
~
~
No write since last change (:quit! overrides)
```

Connected to willow.olemiss.edu

3. *Use the **:q! command** to force quit without saving (**Enter :q!**).*



```
~
~
~
:q!
bash-3.00$
```

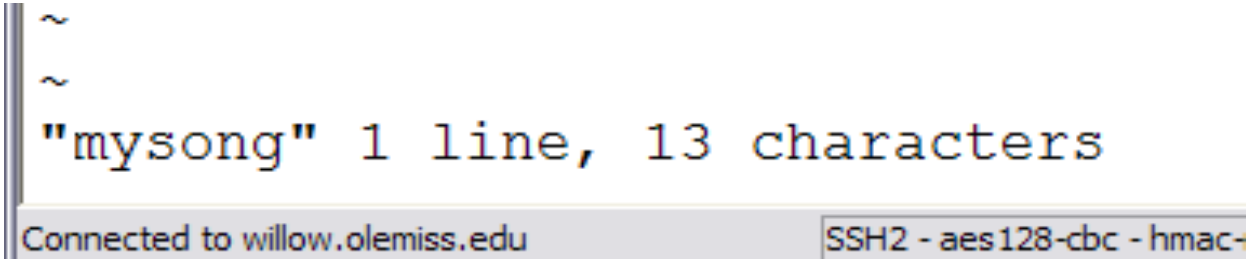
Connected to willow.olemiss.edu

Saving Changes in vi

:wq **Write/save changes and quite**

:*w* **Write/Save changes, but don't quit**

1. *Type **vi mysong** to re-edit your song file.*
2. *Use the **i command** to move into insert mode (**Press i key**).*
3. *Retype the title of your favorite song.*
4. *Use the **Esc key** to move to command mode.*
5. *Use the **:w** command to write/save your edits to file.*



```
~
~
"mysong" 1 line, 13 characters
```

6. *Use the **i command** to enter **insert** mode (**Enter i**).*
7. *Type **Title:** somewhere on the line with the song title.*
8. *Use the **Esc key** to move to command mode.*
9. *Use the **:wq** command to save and quit vi .*

Vi Editor

- **How to type commands in command mode**

  [count] command [where]

  count : Its a number

  where : Specifies how many lines or how much of the document the command affect. It can also be any command that moves the cursor.

Moving the cursor in vi

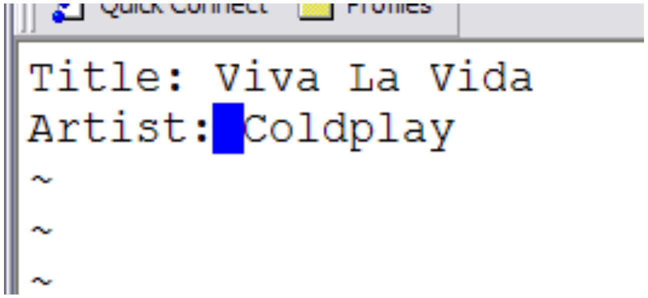
***h* key** **move cursor to the *left* one position**

l key **move cursor right *one* position**

j key **move cursor *down* one line**

k key **move cursor *up* one line**

1. *Type **vi mysong** to re-edit your song file.*
2. *Use the **l command** several times to move cursor to the far right*
3. *Use the **a command** to move into **append** mode (**Press a key**).*
4. *Use the **Enter key** to start a new line of text.*
5. *Type: **Artist:** and then the name of the artist*
6. *Use the **Esc key** to move to command mode .*
7. *Practice moving cursor up, down, left,*
   *and right with **h,l,j,k** keys.*



```
Title: Viva La Vida
Artist: Coldplay
~
~
~
```

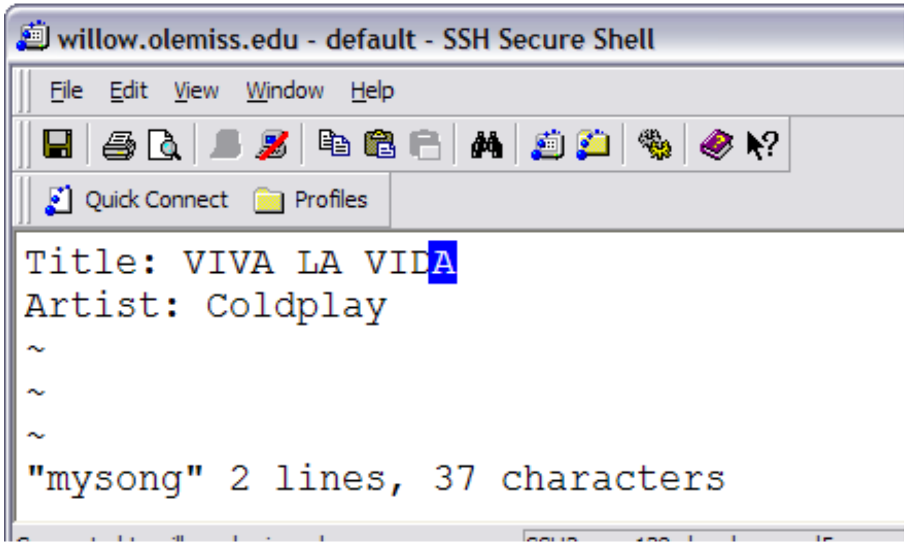
Simple vi editing commands

r **replace one character under the cursor**

x **delete 1 character under the cursor.**

2x **delete 2 characters (3x, etc.)**

u **undo the last change to the file**



```
Title: VIVA LA VIDA
Artist: Coldplay
~
~
~
"mysong" 2 lines, 37 characters
```

1. *Use the **Esc key** to make sure you are still in **command mode**.*
2. *Reposition your cursor and use the **a, l, r** and **x commands** to repair any typos in your title and artist, and change the title to **ALL CAPS***
3. *Use the **:w** command to save your changes.*

Cutting text in Vi

d^

Deletes from current cursor position to the beginning of the line

d$

Deletes from current cursor position to the

end of the line

Dw

Deletes from current cursor position to the

end of the word

dd

Deletes one line from current cursor position. Specify count to delete many lines.

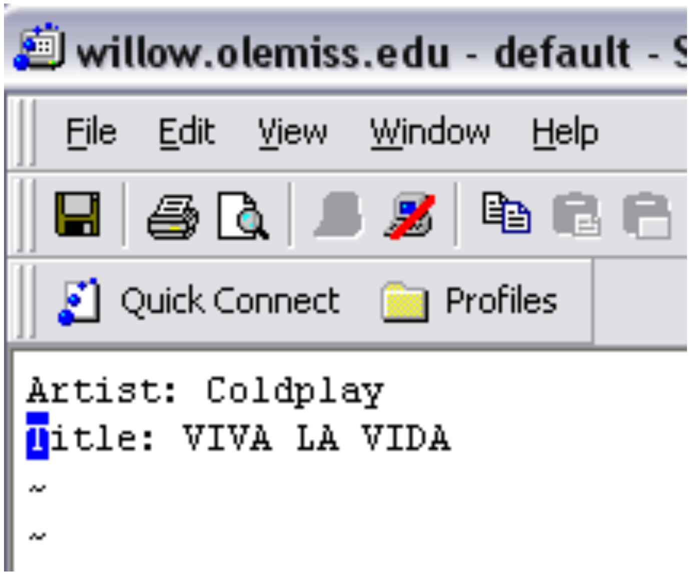
Cutting & Yanking Text in Vi

dd **Delete (cut) 1 line from current cursor position**

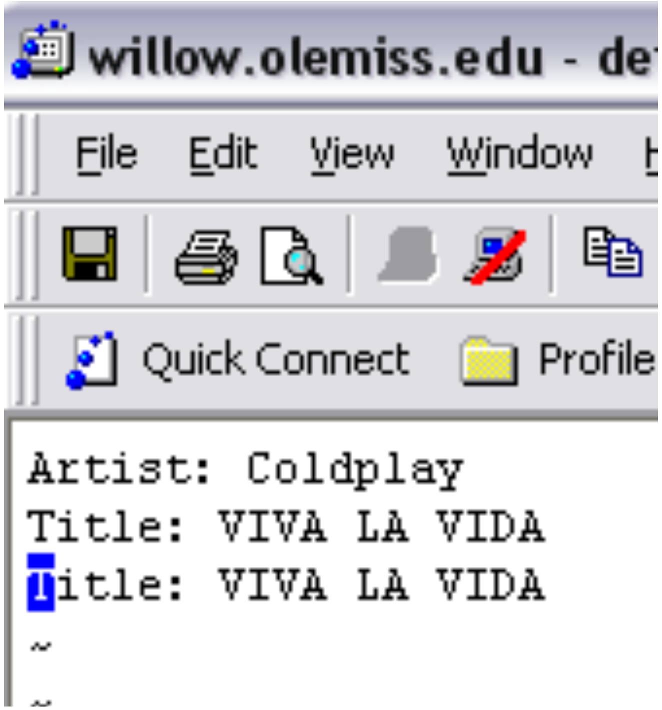
2dd **Delete (cut) 2 lines (*3dd* to cut 2 lines, etc.)**

p **paste lines below current line**

1. *Move cursor to top line and type **dd** to cut the title line*
2. *Use the **p** command to paste the title line below the artist line*
3. *Use the **p** command to paste it again.*



```
Artist: Coldplay
Title: VIVA LA VIDA
~
~
```



```
Artist: Coldplay
Title: VIVA LA VIDA
Title: VIVA LA VIDA
~
~
```

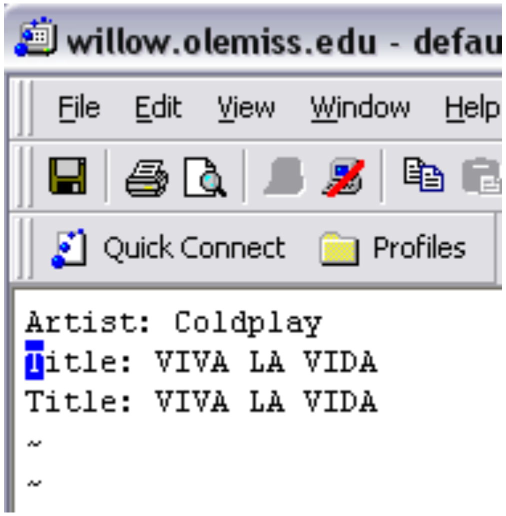
Cutting & Yanking Text in Vi

yy **yank (copy) a single line**

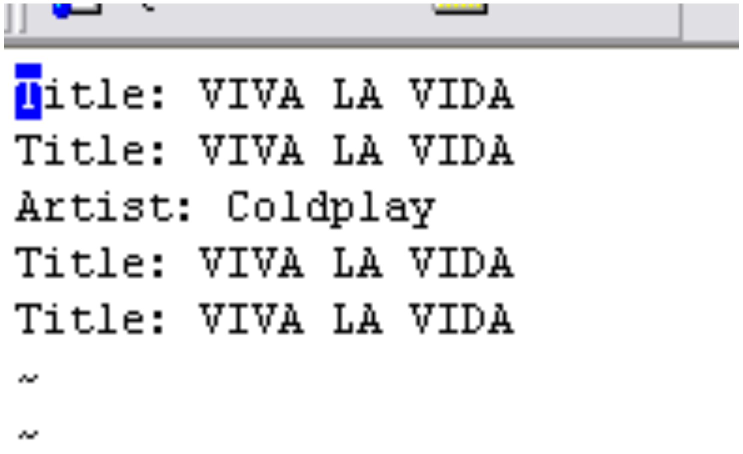
2yy **yank (copy) 2 lines (*3yy* to copy 3 lines, etc.)**

P **paste lines before current line**

1. *Move cursor to first of the 2 title lines and type **2yy** to yank/copy 2 lines*
2. *Move cursor to the first line, then use the **capital P** command to paste the two yanked links above the artist*



```
Artist: Coldplay
Title: VIVA LA VIDA
Title: VIVA LA VIDA
~
~
```



```
Title: VIVA LA VIDA
Title: VIVA LA VIDA
Artist: Coldplay
Title: VIVA LA VIDA
Title: VIVA LA VIDA
~
~
```

Vi Editor

To go to a specific line in the file

:linenumber

1. *Go to the 3rd line by typing* ***:3***
2. *Go to the 1st line by typing* ***:1***
3. *Go to the last line by typing* ***G***

Vi string/search

/[pattern] search forward for the pattern

?[pattern] search backward for the pattern

n   search for the next instance of a string

1. *Search forward for the next line containing the string **Title** by typing **/Title***
2. *Search forward for the next instance of **Title** by typing **n***
3. *Search backward for the most recent instance of Title by typing **?Title***
4. *Search backward for the next most recent instance of Title by typing **n***

More commands

yl

yank a single character. Specify count to yank more characters

yw

yank a single word. Specify count to yank more words

d^

Deletes from current cursor position to the beginning of the line

d$

Deletes from current cursor position to the

end of the line

Dw

Deletes from current cursor position to the

end of the word

THE END
THANK YOU